

Stored Procedures (and Functions) - Simple (PostgreSQL Version)

A **stored procedure** in PostgreSQL is a named block of SQL statements that can be executed as a single unit. Stored procedures allow you to group related logic together, accept parameters, and control program flow; much like procedures in other programming languages.

Characteristics of Stored Procedures

- Stored procedures are **saved inside the database** and can be called repeatedly without re-parsing or re-planning each time (which improves performance).
 - The **source code** of the procedure is stored in the database system catalogs.
 - Once created, a stored procedure can be executed without recompiling the SQL each time.
 - **Important:** Even though the source code is stored in the database, it is good practice to keep an external copy of your procedure code for backup or version control.
-

Simplified Syntax

In PostgreSQL, stored procedures are created using the CREATE PROCEDURE statement. They are written using the **PL/pgSQL** procedural language.

Syntax:

```
CREATE PROCEDURE procedure_name ( [parameter_list] )  
LANGUAGE plpgsql  
AS $$body$$  
BEGIN  
    -- SQL statements  
END;  
$$body$$;
```

Notes:

- The LANGUAGE plpgsql clause specifies that the procedure uses PostgreSQL's procedural language.
 - \$\$body\$\${...}\$\$body\$\$ marks the code block. With \$\$, you don't need to escape single quotes inside the block. The code looks cleaner and easier to read.
 - Unlike SQL Server, PostgreSQL **does not use RETURN** in procedures. Instead, to return values, create a function (using CREATE FUNCTION).
-

Example: Simple Stored Procedure

```
Create Procedure HelloWorld()  
Language plpgsql  
As $$body$$  
Begin  
    Raise Notice 'Hello, World!';  
End;  
$$body$$;
```

Executing the Procedure

To run a procedure, use the CALL command:

```
Call HelloWorld();
```

Stored Procedure Capabilities

A PostgreSQL stored procedure can include:

- **DML statements** (INSERT, UPDATE, DELETE, SELECT)
- **Flow control** (IF, ELSIF, LOOP, etc.)
- **Transactions** (procedures can explicitly COMMIT or ROLLBACK)
- **Exception handling** (BEGIN ... EXCEPTION ... END)
- **Variable declarations and assignments**

Modifying a Stored Procedure

You can update an existing procedure in one of two ways:

1. Drop and Recreate

Dropping removes the procedure completely from the database.

DROP PROCEDURE ***procedure_name*** ([parameter_types]);

Then recreate it using CREATE PROCEDURE.

2. Use ALTER PROCEDURE

You can modify some attributes (such as the owner or schema) or replace the procedure definition using CREATE OR REPLACE PROCEDURE.

CREATE OR REPLACE PROCEDURE ***procedure_name*** ([parameter_list])

LANGUAGE plpgsql

AS \$body\$

BEGIN

 -- Updated SQL statements

END;

\$body\$;

Note: CREATE OR REPLACE PROCEDURE replaces the procedure body (between Begin and End) but does **not** overwrite anything else. If the input arguments are different, PostgreSQL creates a new procedure with the same name and different input arguments.

```
Create Procedure PR_UpdateWithdrawStatus()
```

```
Language plpgsql
```

```
As $body$
```

```
Begin
```

```
    Update      Registration
```

```
    Set         WithdrawYN = 'Y'
```

```
    Where              Mark < 50;
```

```
End;
```

```
$body$;
```

Invoking Stored Procedures

To execute a procedure:

```
CALL procedure_name ( [arguments] );
```

Example:

```
Call PR_UpdateWithdrawStatus();
```

If you need to return a value, you will want to use a function instead. Stored functions are created using the CREATE FUNCTION statement. They are also written using the PL/pgSQL procedural language.

Simplified Syntax:

```
CREATE FUNCTION function_name ( [parameter_list] )
```

```
Returns data_type
```

```
LANGUAGE plpgsql
```

```
AS $$
```

```
BEGIN
```

```
    -- SQL statements
```

```
    Return value;
```

```
END;
```

\$\$;

Example:

Create Function FN_GetTotalPayments()

Returns decimal(8,2)

Language plpgsql

As \$body\$

Declare

v_total decimal(8,2);

Begin

Select Sum(Amount)

Into v_total

From Payments;

Return v_total;

End;

\$body\$;

Invoking Stored Function

To run a function:

Select FN_GetTotalPayments();